AMENDMENT OF CLAIMS

(currently amended)

1. A parallel processing method for performing ~~processing~~ computer graphics shading tasks in parallel on a plurality of processors comprising:

(a) identifying at least one shading task area of a large computer graphics rendering ~~processing~~ task directed to a plurality of ~~computational~~ computer shading task processes for shading an image frame of a scene that can be grouped together as a shading task space not dependent on passing of control of processing from an external process in order to complete processing of the computational processes of the shading task space;

(b) breaking down the shading task space into a plurality of self-contained shading task objects each of which can be executed in one computational step without requiring passing of control to or from another object, wherein each shading task object is defined with a computational step and at least one "data-waiting" slot for receipt of data requested from another shading task object to which the aforesaid task object passes a message for the requested data, and wherein once all the "data-waiting" slots of a shading task object are filled by the respective return message(s), the shading task object can perform its defined computational step without waiting for any other input;

(c) scheduling the defined shading task objects of said identified shading task space so that each shading task object ready for processing is processed by a next available "unoccupied" one of the plurality of processors, by the sequence of:

(i) placing a shading task object with an unfilled "data-waiting" slot in a "waiting" state in which it is not assigned to any processor;

(ii) changing the status of a shading task object to an "active" state when all of its defined "data-waiting" slots have been filled, wherein it is assigned to a next available processor in an "unoccupied" state, then placing that processor's status in an "occupied" state; and

(iii) changing the status of the shading task object to a "dead" state when the computational step to be performed for the task object by the assigned processor has been completed, and then changing the processor's status to an "unoccupied" state to be assigned to a next "active" shading task object.

(currently amended)

2. A parallel processing method according to Claim 1, wherein a master shading task grouping is defined by a plurality of shading task spaces each of which contains multiple shading task objects and does not require passing of control from an external source in order to complete computation for the respective shading task space.

(currently amended)

3. A parallel processing method according to Claim 2, wherein all shading task objects of the shading task spaces which are in an"active" state are placed in a processing queue and each is assigned in turn to a next available "unoccupied" processor.

(currently amended)

4. A parallel processing method according to Claim 3, wherein a master engine for the master shading task grouping maintains threads which track the processing of shading task objects in each of the shading task spaces.

(currently amended)

5. A parallel processing method according to Claim 4, wherein the master engine for the master shading task grouping maintains an internal space address assigned to each respective shading task object.

(currently amended)

6. A parallel processing method according to Claim 5, wherein a shading task object in one master shading task grouping can exchange data with a shading task object in another master shading task grouping by providing its internal space address indexed to its master shading task grouping.

(Claim 7, canceled)

(currently amended)

8. A parallel processing method according to Claim ~~7~~ 1, wherein the shading task

master shading task grouping.

(Claim 7, canceled)

(currently amended)

8. A parallel processing method according to Claim ~~7~~ 1, wherein the shading task includes a master shading task grouping of shading task spaces each of which performs shading of a pixel in the image frame.

(previously presented)

9. A parallel processing method according to Claim 8, wherein each shading task space includes a plurality of "pixel shading" task objects for performing shading of the pixel based upon ray shooting from light sources in the scene, and a "compositing" task object for compositing the shading results for the pixel.

(previously presented)

10. A parallel processing method according to Claim 9, wherein each shading task object has at least one "data-waiting" slot for return of data characterizing light emitted from a respective light source in the scene.

(previously presented)

11. A parallel processing method according to Claim 9, wherein the rendering task includes a function for receiving scene data for a "world map" of the scene, a function for defining the scene objects in each frame of the scene, a function for defining the pixels of an object in the scene intersected by an eye ray of a viewer of the scene, and a function for tiling together the shading results returned by each of the master shading task groupings for respective objects in the image frame.

(currently amended)

12. A software programming method for performing ~~processing~~ computer

task objects each of which can be executed in one computational step without requiring passing of control to or from another object, wherein each shading task object is defined with a computational step and at least one "data-waiting" slot for receipt of data requested from another shading task object to which the aforesaid task object passes a message for the requested data, and wherein once all the "data-waiting" slots of a shading task object are filled by the respective return message(s), the shading task object can perform its defined computational step without waiting for any other input;

(c) scheduling the defined shading task objects of said identified shading task space so that each shading task object ready for processing is processed by a next available "unoccupied" one of the plurality of processors, by the sequence of:

(i) placing a shading task object with an unfilled "data-waiting" slot in a "waiting" state in which it is not assigned to any processor;

(ii) changing the status of a shading task object to an "active" state when all of its defined "data-waiting" slots have been filled, wherein it is assigned to a next available processor in an "unoccupied" state, then placing that processor's status in an "occupied" state; and

(iii) changing the status of the shading task object to a "dead" state when the computational step to be performed for the shading task object by the assigned processor has been completed, and then changing the processor's status to an "unoccupied" state to be assigned to a next "active" shading task object.

(currently amended)

13. A software programming method according to Claim 12, wherein a master shading task grouping is defined by a plurality of shading task spaces each of which contains multiple shading task objects and does not require passing of control from an external source in order to complete computation for the respective shading task space.

(currently amended)

14. A software programming method according to Claim 13, wherein all shading task objects of the shading task spaces which are in an"active" state are placed in a processing queue and each is assigned in turn to a next available "unoccupied" processor.

(currently amended)

15. A software programming method according to Claim 14, wherein a master engine for the master shading task grouping maintains threads which track the processing of shading task objects in each of the shading task spaces.

(currently amended)

16. A software programming method according to Claim 15, wherein the master engine for the master shading task grouping maintains an internal space address assigned to each respective shading task object.

(currently amended)

17. A software programming method according to Claim 16, wherein a shading task object in one master shading task grouping can exchange data with a shading task object in another master shading task grouping by providing its internal space address indexed to its master shading task grouping.

(previously presented)

18. A software programming method according to Claim 12, further comprising storing templates for different types of task engines, spaces, and objects in a library and utilizing the templates to generate software programming for a desired processing task.

(Claim 19, canceled)

(currently amended)

20. A software programming method according to Claim ~~19~~ 12, wherein the shading task processes includes a master task grouping of shading task spaces each of which performs shading of a pixel in the image frame.